



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/047,312	01/14/2002	Kevin S. Barker	RSW920010187US1	5177
48816	7590	04/06/2006		
VAN LEEUWEN & VAN LEEUWEN P.O. BOX 90609 AUSTIN, TX 78709-0609			EXAMINER MITCHELL, JASON D	
			ART UNIT	PAPER NUMBER
			2193	

DATE MAILED: 04/06/2006

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

**MAILED**

**APR 6 2006**

**Technology Center 2100**

**BEFORE THE BOARD OF PATENT APPEALS  
AND INTERFERENCES**

Application Number: 10/047,312  
Filing Date: January 14, 2002  
Appellant(s): BARKER ET AL.

\_\_\_\_\_  
Joseph T. Van Leeuwen  
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 1/10/06 appealing from the Office action  
mailed 8/11/05.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

Copending US application 10/046,940 is currently under appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

US 6,311,321 to Agnihotri et al. 10/2001

"Common Information Model (CIM) Specification v. .2.2" 5/1999

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

Art Unit: 2193

Claims 1, 3-8, 10-15, 17-21 and 25 are rejected under 35 U.S.C. 102(e) as being anticipated by US 6,311,321 to Agnihotri et al. (Agnihotri).

Claims 2, 9, 16 and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,311,321 to Agnihotri et al. (Agnihotri) in view of "Common Information Model (CIM) Specification v. 2.2" (CIM)

**Claims 1, 3-8, 10-15, 17-21 and 25 are rejected under 35 U.S.C. 102(e) as being anticipated by US 6,311,321 to Agnihotri et al. (Agnihotri).**

**Regarding Claims 1, 8 and 15:** Agnihotri discloses receiving one or more console identifiers, each of the console identifiers corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); retrieving one or more plug-in code files, each of the plug-in code files derived from the management data (col. 4, lines 50-55 'applet components') and each adapted to interface with one of the management consoles (col. 4, lines 50-55 'specific to the Enterprise management console'); retrieving one or more display panel files derived from the management data (col. 5, lines 10-12 'The install file may contain a ... image file for graphical representation on the console'); and writing the plug-in code files and the display panels to a distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium').

As discussed in more detail in the Arguments section of this document, any application written to manage a device is necessarily 'derived from management data' related to that device. While it is acknowledged that Agnihotri does not explicitly disclose a

derivation process, to the extent recited in the claims, such a process is inherent in the nature of Agnihotri's 'applet'.

**Regarding Claims 3, 10 and 17:** The rejection of claims 1, 8 and 15 are incorporated respectively; further Agnihotri discloses retrieving one or more translation files derived from the management data (col. 4, 50-55 'class object definitions'), each of the translation files corresponding to at least one national language (col. 5, lines 43-46 'console information such as ... language'); and writing the translation files to the distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium').

**Regarding Claims 4, 11 and 18:** The rejection of claims 1, 8 and 15 are incorporated respectively; further Agnihotri discloses that each of the display panel files is adapted to operate with a plurality of the management consoles (col. 4, lines 58-63 'generic interface').

Also see Fig. 3, where as discussed in the Arguments section, the plurality of console install DLLs 216A-216C represent an adaptation mechanism contained in the "Install framework 212" which is used to 'integrate the component (applet)' (col. 5, lines 8-10 'the component (applet) to be installed') including the associated 'image file for graphical representation on the console' (col. 5, lines 10-12)

**Regarding Claims 5, 12 and 19:** The rejection of claims 1, 8 and 15 are incorporated respectively; further Agnihotri discloses retrieving one or more plug-in runtime algorithms (col. 7, lines 3-8 'interface COM objects'), each of the algorithms corresponding to one of the console identifiers (col. 7, lines 3-8 'console specific

commands'); generating a console plug-in code file for each of the console identifiers (col. 7, lines 3-8 'COM objects ... translate instructions ... into console specific commands'; Fig. 4); the compiling resulting in an executable entity adapted to interface with the management console corresponding to the console identifier (col. 6, lines 49-53 'a comprehensive generic interface to existing Enterprise management consoles') and inherently discloses compiling each of the generated console plug-in files (Fig. 4). Additionally, as discussed in more detail in the arguments section, it can be seen from Agnihotri's Fig. 4 that the 'MConsole' and 'COM objects' are linked thereby 'generating a plug-in code file'. Additionally such a step would first require that the objects (COM objects in particular) be retrieved from whatever storage medium they were residing on. Further, Fig. 4 shows that these objects are in DLL and COM formats, respectively, and thus must have been compiled to produce executable objects.

**Regarding Claims 6, 13 and 20:** The rejection of claims 1, 8 and 15 are incorporated respectively; further Agnihotri discloses loading the distribution medium into a computer system (col. 4, lines 47-48 'install module'); displaying a name corresponding to each of the management consoles in a selection display (col. 5, lines 3-7 'provides the user the option to select one console'); receiving one or more selections from a user, each of the selections corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); copying the plug-in code files corresponding to the selected management consoles from the distribution medium to a nonvolatile storage device accessible by the computer system (col. 5, lines 15-18 'install the component'); copying the display panel files from the distribution medium to a

Art Unit: 2193

nonvolatile storage device accessible by the computer system (col. 5, lines 15-18 'may proceed to install the component'); and registering each of the plug-in code files with one or more installed management consoles (col. 5, lines 15-18 'and make appropriate updates to the console to integrate the component'), wherein the installed management consoles are installed on the computer system (col. 5, lines 5-6 'consoles are installed at a host system').

**Regarding Claims 7, 14 and 21:** The rejection of claims 6, 13 and 20 are incorporated respectively; further Agnihotri discloses invoking one of the installed management consoles (col. 5, lines 15-18 'and make appropriate updates to the console to integrate the component'); receiving a console selection from a user (col. 5, lines 3-7 'provides the user the option to select one console'); and displaying a display panel corresponding to one of the display panel files in response to the received selection (col. 5, lines 10-12 'an image file for graphical representation on the console').

**Regarding Claim 25:** Agnihotri discloses a computer program product stored on a computer operable medium for packaging management data adapted to interoperate with one or more management consoles, said computer program product comprising: means for receiving one or more console identifiers, each of the console identifiers corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); means for retrieving one or more plug-in code files (col. 4, lines 50-55 'applet components'), each of the plug-in code files derived from the management data and each adapted to interface with one of the management consoles (col. 4, lines 50-55 'specific to the Enterprise management console'); means for

Art Unit: 2193

retrieving one or more display panel files derived from the management data (col. 5, lines 10-12 'The install file may contain a ... image file for graphical representation on the console'); means for retrieving one or more translation files derived from the management data (col. 4, 50-55 'class object definitions'), each of the translation files corresponding to at least one national language (col. 5, lines 43-46 'console information such as ... language'); and means for writing the translation files the plug-in code files and the display panels to a distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium').

**Claims 2, 9, 16 and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,311,321 to Agnihotri et al. (Agnihotri) in view of "Common Information Model (CIM) Specification v. 2.2" (CIM).**

**Regarding Claims 2, 9 and 16:** The rejection of claims 1, 8 and 15 are incorporated respectively; further Agnihotri discloses the use of non-console specific management applications (col. 6, line 56), but does not disclose that the definition is a common information model managed object format.

CIM teaches a common information model managed object format file (pg. 1, ch. 1.1 'management schemas') in an analogous art for the purpose of providing a 'conceptual framework ... to organize the available information about the managed environment' (pg. 1, ch. 1.1).

It would have been obvious to a person of ordinary skill in the art at the time of the invention to model the 'device management applications' disclosed in Agnihotri (col. 6,



line 56) using the CIM management schema because one of ordinary skill in the art would have been motivated to model the management information using a method which allows for ease of development and reuse (CIM Ch. 1 Introduction and Overview 'Ideally, information used to perform tasks is organized or structured to allow disparate groups of people to use it')

**Regarding Claim 22:** Agnihotri discloses a method of packaging management data adapted to interoperate with one or more management consoles, said method comprising: receiving one or more console identifiers, each of the console identifiers corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); retrieving one or more plug-in code files, each of the plug-in code files derived from the management data and each adapted to interface with one of the management consoles (col. 4, lines 50-55 'applet components'); retrieving one or more display panel files derived from the management data (col. 5, lines 10-12 'The install file may contain a ... image file for graphical representation on the console'); retrieving one or more translation files derived from the management data, each of the translation files corresponding to at least one national language (col. 5, lines 43-46 'console information such as ... language'); and writing the translation files, the plug-in code files and the display panels to a distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium'). Agnihotri does not disclose that the management data includes a common information model managed object format file, but discloses the use of non-console specific management applications (col. 6, line 56).

CIM teaches a common information model managed object format file (pg. 1, ch. 1.1 'management schemas') in an analogous art for the purpose of providing a 'conceptual framework ... to organize the available information about the managed environment'. It would have been obvious to a person of ordinary skill in the art at the time of the invention to model the 'device management applications' disclosed in Agnihotri (col. 6, line 56) using the CIM management schema because one of ordinary skill in the art would have been motivated to model the management information using a method which allows for ease of development and reuse (CIM Ch. 1 Introduction and Overview 'Ideally, information used to perform tasks is organized or structured to allow disparate groups of people to use it').

**Regarding Claim 23:** Agnihotri discloses a method of packaging management data adapted to interoperate with one or more management consoles, said method comprising: receiving one or more console identifiers, each of the console identifiers corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); retrieving one or more plug-in code files, each of the plug-in code files derived from the management data and each adapted to interface with one of the management consoles (col. 4, lines 50-55 'applet components'); retrieving one or more display panel files derived from the management data (col. 5, lines 10-12 'The install file may contain a ... image file for graphical representation on the console'); retrieving one or more translation files derived from the management data, each of the translation files corresponding to at least one national language (col. 5, lines 43-46 'console information such as ... language'); retrieving one or more plug-in runtime

algorithms, each of the algorithms corresponding to one of the console identifiers (col. 6, lines 55-60 'device management applications'); generating a console plug-in code file for each of the console identifiers (col. 7, lines 3-8 'COM objects ... translate instructions ... into console specific commands'); compiling each of the generated console plug-in files, the compiling resulting in an executable entity adapted to interface with the management console corresponding to the console identifier (col. 6, lines 62-65 'The console libraries'); writing the translation files, the compiled plug-in code files and the display panels to a distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium'). Agnihotri does not disclose that the management data includes a common information model managed object format file, but discloses the use of non-console specific management applications (col. 6, line 56).

CIM teaches a common information model managed object format file (pg. 1, ch. 1.1 'management schemas') in an analogous art for the purpose of providing a 'conceptual framework ... to organize the available information about the managed environment'. It would have been obvious to a person of ordinary skill in the art at the time of the invention to model the 'device management applications' disclosed in Agnihotri (col. 6, line 56) using the CIM management schema because one of ordinary skill in the art would have been motivated to model the management information using a method which allows for ease of development and reuse (CIM Ch. 1 Introduction and Overview 'Ideally, information used to perform tasks is organized or structured to allow disparate groups of people to use it').

**Regarding Claim 24:** Agnihotri discloses an information handling system comprising: one or more processors; a memory accessible by the processors; a nonvolatile storage area accessible by the processors; and a packaging tool for packaging management data adapted to interoperate with one or more management consoles, the packaging tool including: input logic for receiving one or more console identifiers, each of the console identifiers corresponding to one of the management consoles (col. 5, lines 3-7 'provides the user the option to select one console'); retrieval logic for retrieving one or more plug-in code files, each of the plug-in code files derived from the management data and each adapted to interface with one of the management consoles (col. 4, lines 50-55 'applet components'); retrieval logic for retrieving one or more plug-in runtime algorithms, each of the algorithms corresponding to one of the console identifiers (col. 6, lines 55-60 'device management applications'); code generation logic for generating a console plug-in code file for each of the console identifiers (col. 7, lines 3-8 'COM objects ... translate instructions ... into console specific commands'); and compiler logic for compiling each of the generated console plug-in files, the compiling resulting in an executable entity adapted to interface with the management console corresponding to the console identifier (col. 6, lines 62-65 'The console libraries'); retrieval logic for retrieving one or more display panel files derived from the management data (col. 5, lines 10-12 'The install file may contain a ... image file for graphical representation on the console'); output logic for writing the compiled plug-in code files and the display panels to a distribution medium (col. 4, lines 32-35 'a software module provided on a tangible medium'). Agnihotri does not disclose that the management data includes a

Art Unit: 2193

common information model managed object format file, but discloses the use of non-console specific management applications (col. 6, line 56).

CIM teaches a common information model managed object format file (pg. 1, ch. 1.1 'management schemas') in an analogous art for the purpose of providing a 'conceptual framework ... to organize the available information about the managed environment'.

It would have been obvious to a person of ordinary skill in the art at the time of the invention to model the 'device management applications' disclosed in Agnihotri (col. 6, line 56) using the CIM management schema because one of ordinary skill in the art would have been motivated to model the management information using a method which allows for ease of development and reuse (CIM Ch. 1 Introduction and Overview 'Ideally, information used to perform tasks is organized or structured to allow disparate groups of people to use it').

## **(10) Response to Argument**

### **I. 35 U.S.C. 102, Anticipation**

#### ***1. Independent Claims 1, 8, 15 and 25***

Starting in the last paragraph on pg. 8, Appellant states:

The rejected independent claims ... all recite "plug-in code files derived from the management data" and "display panel files derived from the management data" ... These features are not taught or suggested by Agnihotri. While the examiner has cited various excerpts from Agnihotri that the Examiner argues teach these claim elements, Agnihotri fails to teach or otherwise describe or suggest the origin of the "applet components" and "install interface" the Examiner refers to.

Examiner respectfully disagrees. As stated in previously, the language of the claims places no limitations on the terms 'management data' and 'derived'. Further, it is noted

Art Unit: 2193

that the 'means plus function' language of claims 15 and 25 does not apply to the derivation aspect of the respective limitations. Consequently the scope of the claims is very broad.

With that in mind Agnihotri's discloses 'device management applications (applets)' (col. 3, lines 24-29) and that applets have associated display files (col. 5, lines 8-12 'an install file for the component (applet) ... may contain ... an image file for graphical representation on the console').

Any application written to manage a device is necessarily 'derived from management data' related to that device. In other words, the application would be unable to interface with the device to perform management functions applicable to that device if it were written without knowledge of (not derived from) the management functions the device is capable of (Management Data). While it is acknowledged that Agnihotri does not explicitly disclose a derivation process, to the extent recited in the claims, such a process is inherent in the nature of Agnihotri's 'applet'.

In the paragraph bridging pp. 9 and 10 Appellant states:

The present invention, on the other hand, is specifically directed to the derivation of plug-in code and display panels from management data. In a preferred embodiment, these plug-in code files and display panel files are specifically derived from management definition objects, such as from an MOF (Management Object Format) file

With the exception of claims 2, 9, 16 and 22-24 which were rejected as unpatentable over Agnihotri in view of CIM, Appellant's asserted limitations (derivation of plug-in code files from an MOF) do not appear in the claims. Further, claims 2, 9, 16 and 22-24, only

recite 'the management data including a common information model and do not recite any limitation specifically regarding the process of derivation.

In the last paragraph on pg. 10 Appellant states:

In addition, with respect to independent claim 25, Agnihotri fails to teach or suggest the claimed feature of "retrieving one or more translation files derived from the management data, each of the translation files *corresponding to at least one national language*" ... Although the Examiner cites an excerpt from Agnihotri that makes a passing reference to "language," the excerpt makes no mention of retrieving any translation files corresponding to any national language. In fact, from the context, it appears that by "language," Agnihotri is referring to a computer programming language, such as C or Pascal, rather than a "national" or "nature" language use for human communication. (emphasis in original)

Examiner respectfully disagrees. The relevant disclosure from Agnihotri is reproduced below.

The Install Interface DLL 214 may store generic instruction sets (i.e., interface programs) for supporting the installation process, including providing console information such as installation directory, language. Console version, etc. Individual instruction sets (in bold print) and textual comments may be written in any of the C-family (e.g., C or C++) code language. However, other program languages included in the non-exhaustive list of Basic and Pascal may also be used.

The cited section of Agnihotri is discussing an "Install Interface", and the reference to "language" is presented as "console information" "for supporting the installation process". Thus it can be seen that 'language' in this context refers to the 'national language' displayed on the 'console' and . Further, Agnihotri's reference to computer programming languages is discussing the implementation of the 'generic instruction sets' and not a console language.

## **2. Dependent Claims Rejected Under 102**

### **2a. Dependent Claims 3, 10 and 17**

Art Unit: 2193

At the bottom of pg. 11, Appellant presents arguments regarding claims 3, 10 and 17 which are essentially the same as those discussed above with regard to claim 25, and consequently are not addressed here.

2b Dependent Claims 4, 11 and 18

Starting in the 2<sup>nd</sup> to last paragraph on pg. 12 Appellant states:

Agnihotri does not teach or suggest a display panel that is adapted to operate with a plurality of management consoles. Instead, Agnihotri teaches separate display interfaces for use with various management consoles.

...

Agnihotri specifically teaches that the "plurality of console install DLLs 216A, 216B, and 216C" are used to provide the interfaces for installation. As shown in Agnihotri's Figure 3, these DLLs are each directed towards a different management console. Consequently, Agnihotri teaches that the display panels are coded in the separate DLLs and are not "adapted to operate with a plurality of the management consoles,"

Examiner respectfully disagrees. Looking to Fig. 3, it can be seen that the "plurality of console install DLLs" (216A-216B) represent an adaptation mechanism contained in the "Install framework 212" which is used to 'integrate the component (applet)' (col. 5, lines 8-10 'the component (applet) to be installed) including an associated 'image file for graphical representation on the console' (col. 5, lines 10-12). Thus it should be clear that Agnihotri discloses a display panel (the 'graphical representation' of a particular applet) that is adapted to operate with a plurality of management consoles (through the use of console install DLLs 216A-C).

2c Dependent Claims 5, 12, and 19

Beginning in the paragraph bridging pp. 13 and 14, Appellant states:

Agnihotri does not teach or suggest the limitations of claims 5, 12, and 19.

...

In the cited section, Agnihotri is discussing interface COM objects that are used to translate instructions from one console (the MConsole Interface) to other consoles.



Nowhere does Agnihotri teach or suggest “retrieving ... plug-in runtime algorithms” that correspond to one of the consoles, as taught and claimed by Appellants. Moreover, Agnihotri does not teach “compiling ... the generated console plug-in files ... resulting in an executable ... adapted to interface with the management console...” Instead, as discussed above, Agnihotri is simply teaching use of a module that translates console instructions from one console to another console.

In col. 6, lines 49-53 Agnihotri discloses “The MConsole Interface module 220 ... [provides] a comprehensive generic interface to existing Enterprise management consoles”. This clearly discloses an executable entity adapted to interface with the management console.

In col. 6, lines 53-61 Agnihotri discloses “COM objects ... for automating integration ... into existing Enterprise management consoles”. This clearly discloses algorithms corresponding a console.

Further it can be seen from Agnihotri’s Fig. 4 that the ‘MConsole’ and ‘COM objects’ are linked thereby ‘generating a plug-in code file’. Additionally such a step would first require that the objects (COM objects in particular) be retrieved from whatever storage medium they were residing on. Further, Fig. 4 shows that these objects are in DLL and COM formats, respectively, and thus must have been compiled to produce executable objects.

#### 2d. Dependent Claims 6, 13 and 20

In the first full paragraph on pg. 15, Appellant states:

The Final Office Action asserts that Agnihotri teaches Appellants’ limitation of “copying the display panel files from the distribution medium to a nonvolatile storage device accessible by the computer system,” citing col. 5, lines 15-18. However, this section of Agnihotri clearly only discusses installing the “applet” (code) and never teaches or suggests installing a separate display panel:

Examiner respectfully disagrees. In col. 5, lines 8-13 Agnihotri discloses that a file associated with each applet ('an install file for the ... applet') 'may contain ... an image file for graphical representation on the console'. Further this 'install file' is clearly included in the install package ('The wizard ... may read an install file') and consequently would have been 'copied from the distribution medium to a nonvolatile storage device'.

2e. Dependent Claims 7, 14 and 21

In the paragraph bridging pp. 15 and 16 Appellant states:

The Final Office Action contends that Appellants last limitation (displaying a display panel corresponding to one of the display panel files in response to the received selection) is taught by Agnihotri at col. 5, lines 10-12. however, the section of Agnihotri merely states that an "install file may contain a configuration file and an image file for graphical representation on the console." Importantly, neither this section, for anywhere else in Agnihotri, teach or suggest displaying a display panel that is separate and apart from the program code as taught and claimed by Appellants.

The relevant passage of Agnihotri is reproduced below:

The wizard application may read an install file for the component (applet) to be installed. The install file may contain a configuration file and an image file for graphical representation on the console.

Appellant does not point out any distinction between the cited passage of Agnihotri and Appellant's claimed displaying a display panel ('an image file for graphical representation on the console') that is separate and apart from the program code ('an install file for the component (applet)').

Representing an image file on the console falls well within the scope of "displaying a display panel", and it is clear that 'an install file for a component is not the component itself and consequently is separate and apart from the component. Therefore Agnihotri's

Art Unit: 2193

disclosed 'image file' (col. 5, lines 10-12) reads on the claimed feature of 'a display panel that is separate and apart from the program code'.

## **II. 35 U.S.C. 103, Alleged Obviousness**

Appellant's arguments on pp. 16-18 regarding the 103 obviousness rejection of claims 2, 9, 16 and 22-24 rely on arguments made against the rejection of previous claims, which have been addressed above, and consequently are not addressed here.

Further, Applicant has not argued why it would not have been obvious to combine the teachings of Agnihotri with those of the CIM specification, as presented in the rejections above. Accordingly this combination is considered proper.

Art Unit: 2193

**(11) Evidence Appendix**

There is no additional evidence entered and relied upon in this appeal

**(12) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Jason Mitchell

3/30/06




Conferees:

Kakali Chaki



**KAKALI CHAKI  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100**

Tuan Dam



**TUAN DAM  
SUPERVISORY PATENT EXAMINER**

Distributed Management Task Force, Inc.



## **Specification**

DSP0004

---

# **COMMON INFORMATION MODEL (CIM) SPECIFICATION**

---

Version 2.2

June 14, 1999

Technical inquiries and editorial comments should be directed in writing to:

Distributed Management Task Force, Inc. (DMTF)  
c/o MacKenzie Kesselring, Inc.  
200 SW Market Street, Suite 450  
Portland, OR 97201  
(503) 294-0739  
(503) 225-0765 (fax)

email: [dmtf-info@dmtf.org](mailto:dmtf-info@dmtf.org)

Additional electronic copies of this specification can be obtained free of charge from the Internet at:

<ftp://ftp.dmtf.org>

or

from the World Wide Web at:

<http://www.dmtf.org>

Additional hardcopies can be obtained for a fee by contacting the DMTF at the address listed above.

#### IMPORTANT INFORMATION AND DISCLAIMERS

1. THIS SPECIFICATION (WHICH SHALL INCORPORATE ANY REVISIONS, UPDATES, AND MODIFICATIONS HERETO) IS FURNISHED FOR INFORMATIONAL PURPOSES ONLY. INTEL CORPORATION, MICROSOFT CORPORATION, DIGITAL EQUIPMENT CORPORATION, HEWLETT-PACKARD COMPANY, INTERNATIONAL BUSINESS MACHINES CORPORATION, NOVELL INC., SUN MICROSYSTEMS, INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORP., SYMANTEC, THE SANTA CRUZ OPERATION, NEC TECHNOLOGIES, INC., OR ANY OTHER DMTF MEMBER MAKE NO WARRANTIES WITH REGARD THERETO, AND IN PARTICULAR DO NOT WARRANT OR REPRESENT THAT THIS SPECIFICATION OR ANY PRODUCTS MADE IN CONFORMANCE WITH IT WILL WORK IN THE INTENDED MANNER OR BE COMPATIBLE WITH OTHER PRODUCTS IN NETWORK SYSTEMS. NOR DO THEY ASSUME RESPONSIBILITY FOR ANY ERRORS THAT THE SPECIFICATION MAY CONTAIN OR HAVE ANY LIABILITIES OR OBLIGATIONS FOR DAMAGES INCLUDING, BUT NOT LIMITED TO, SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, OR CONSEQUENTIAL DAMAGES WHETHER ARISING FROM OR IN CONNECTION WITH THE USE OF THIS SPECIFICATION IN ANY WAY. CORPORATIONS MAY FOLLOW OR DEVIATE FROM THIS SPECIFICATION AT ANY TIME.

2. NO REPRESENTATIONS OR WARRANTIES ARE MADE THAT ANY PRODUCT BASED IN WHOLE OR IN PART ON THE ABOVE SPECIFICATION WILL BE FREE FROM DEFECTS OR SAFE FOR USE FOR ITS INTENDED PURPOSE. ANY PERSON MAKING, USING OR SELLING SUCH PRODUCT DOES SO AT HIS OWN RISK.

3. THE USER OF THIS SPECIFICATION HEREBY EXPRESSLY ACKNOWLEDGES THAT THE SPECIFICATION IS PROVIDED AS IS, AND THAT THE DMTF, NEITHER INDIVIDUALLY NOR COLLECTIVELY, MAKE ANY REPRESENTATIONS, EXTEND ANY WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, ORAL OR WRITTEN, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTY OR REPRESENTATION THAT THE SPECIFICATION OR ANY PRODUCT OR TECHNOLOGY UTILIZING ANY ASPECT OF THE SPECIFICATION WILL BE FREE FROM ANY CLAIMS OF INFRINGEMENT OF INTELLECTUAL PROPERTY, INCLUDING PATENTS, COPYRIGHT AND TRADE SECRETS OF ANY THIRD PARTY, OR ASSUMES ANY OTHER RESPONSIBILITIES WHATSOEVER WITH RESPECT TO THE SPECIFICATION OR SUCH PRODUCTS. IN NO EVENT WILL DMTF MEMBERS BE LIABLE FOR ANY LOSSES, DAMAGES INCLUDING, WITHOUT LIMITATION, THOSE DAMAGES DESCRIBED IN SECTION 1 ABOVE, COSTS, JUDGMENTS, OR EXPENSES ARISING FROM THE USE OR LICENSING OF THE SPECIFICATION HEREUNDER.

## Abstract

The DMTF Common Information Model (CIM) is an approach to the management of systems and networks that applies the basic structuring and conceptualization techniques of the object-oriented paradigm. The approach uses a uniform modeling formalism that—together with the basic repertoire of object-oriented constructs—supports the cooperative development of an object-oriented schema across multiple organizations.

A management schema is provided to establish a common conceptual framework at the level of a fundamental typology—both with respect to classification and association, and with respect to a basic set of classes intended to establish a common framework for a description of the managed environment. The management schema is divided into these conceptual layers:

- Core model—an information model that captures notions that are applicable to all areas of management.
- Common model—an information model that captures notions that are common to particular management areas, but independent of a particular technology or implementation. The common areas are systems, applications, databases, networks and devices. The information model is specific enough to provide a basis for the development of management applications. This model provides a set of base classes for extension into the area of technology-specific schemas. The Core and Common models together are expressed as the CIM schema.
- Extension schemas—represent technology-specific extensions of the Common model. These schemas are specific to environments, such as operating systems (for example, UNIX<sup>†</sup> or Microsoft Windows<sup>†</sup>).

## PARTICIPANTS

This list shows the names of the companies that participated in the Distributed Management Task Force - CIM Sub-Committee whose contributions made this document possible.

- Compaq Computer Corporation
- Computer Associates Intl., Inc
- Hewlett-Packard Company
- Intel Corporation
- Microsoft Corporation
- Novell, Inc.
- Sun Microsystems, Inc.
- Tivoli Systems, Inc.

---

<sup>†</sup> Other product and corporate names may be trademarks of other companies and are used only for explanation and to the owners' benefit, without intent to infringe.

## Change History

Version 1	Wednesday, April 09, 1997	First Public Release
Version 1.1	Thursday, October 23, 1997	Output after Working Groups input
Version 1.2a	Monday, November 03, 1997	Naming
Version 1.2b	Monday, November 17, 1997	Remove reference qualifier
Version 2.0a	Thursday, December 11, 1997	Apply pending changes and new metaschema
Version 2.0d	Thursday, December 11, 1997	Output of 12/9/1997 TDC, Dallas
Version 2.0f	Monday, February 16, 1998	Output of 2/3/1998 TDC, Austin
Version 2.0g	Thursday, February 26, 1998	Apply approved change requests and final edits submitted through 2/26/1998.
Version 2.2	Tuesday, June 14, 1999	Incorporate Errata and approved change requests through 1999-06-08



## Contents

<b>1</b>	<b>INTRODUCTION AND OVERVIEW .....</b>	<b>1</b>
1.1	CIM Management Schema .....	1
1.1.1	Core Model.....	2
1.1.2	Common Model.....	2
1.1.3	Extension Schema .....	2
1.2	CIM Implementations .....	2
1.2.1	CIM Implementation Conformance.....	4
<b>2</b>	<b>META SCHEMA.....</b>	<b>5</b>
2.1	Definition of the Meta Schema .....	5
2.2	Property Data Types .....	10
2.2.1	Date, Time, and Interval Types .....	11
2.2.2	Indicating Additional Type Semantics with Qualifiers .....	12
2.3	Supported Schema Modifications .....	12
2.3.1	Schema Versions .....	13
2.4	Class Names .....	13
2.5	Qualifiers.....	14
2.5.1	Meta Qualifiers.....	14
2.5.2	Standard Qualifiers.....	14
2.5.3	Optional Qualifiers .....	21
2.5.4	User-defined Qualifiers .....	24
2.5.5	Mapping MIF Attributes.....	24
2.5.6	Mapping Generic Data to CIM Properties.....	25
<b>3</b>	<b>MANAGED OBJECT FORMAT .....</b>	<b>27</b>
3.1	MOF usage .....	27
3.2	Class Declarations .....	27
3.3	Instance Declarations .....	27
<b>4</b>	<b>MOF COMPONENTS .....</b>	<b>29</b>
4.1	Keywords.....	29
4.2	Comments.....	29
4.3	Validation Context.....	29

4.4	Naming of Schema Elements .....	29
4.5	Class Declarations .....	30
4.5.1	Declaring a Class .....	30
4.5.2	Subclasses .....	31
4.5.3	Default Property Values .....	31
4.5.4	Class and Property Qualifiers .....	31
4.5.5	Key Properties .....	34
4.6	Association Declarations .....	35
4.6.1	Declaring an Association .....	35
4.6.2	Subassociations .....	35
4.6.3	Key References and Properties .....	36
4.6.4	Object References .....	36
4.7	Qualifier Declarations .....	37
4.8	Instance Declarations .....	37
4.8.1	Instance Aliasing .....	38
4.8.2	Arrays .....	39
4.9	Method Declarations .....	40
4.10	Compiler Directives .....	41
4.11	Value Constants .....	42
4.11.1	String Constants .....	42
4.11.2	Character Constants .....	43
4.11.3	Integral Constants .....	43
4.11.4	Floating-Point Constants .....	43
4.11.5	Object Ref Constants .....	44
4.11.6	NULL .....	44
4.12	Initializers .....	44
4.12.1	Initializing Arrays .....	44
4.12.2	Initializing References Using Aliases .....	45
5	NAMING .....	46
5.1	Background .....	46
5.1.1	Management Tool Responsibility for an Export Operation .....	49
5.1.2	Management Tool Responsibility for an Import Operation .....	49
5.2	Weak Associations: Supporting Key Propagation .....	49
5.2.1	Referencing Weak Objects .....	51
5.3	Naming CIM Objects .....	51
5.3.1	Namespace Path .....	52
5.3.2	Model Path .....	53
5.3.3	Specifying the Object Name .....	54
5.4	Specifying Object Names in MOF Files .....	55
5.4.1	Synchronizing Namespaces .....	55
5.4.2	Building References Between Management Systems .....	58

<b>6</b>	<b>MAPPING EXISTING MODELS INTO CIM.....</b>	<b>61</b>
6.1	Technique Mapping.....	61
6.2	Recast Mapping .....	62
6.3	Domain Mapping.....	64
6.4	Mapping Scratch Pads .....	65
<b>7</b>	<b>REPOSITORY PERSPECTIVE .....</b>	<b>66</b>
7.1	DMTF MIF Mapping Strategies .....	67
7.2	Recording Mapping Decisions.....	68
	<b>APPENDIX A MOF SYNTAX GRAMMAR DESCRIPTION.....</b>	<b>71</b>
	<b>APPENDIX B CIM META SCHEMA .....</b>	<b>76</b>
	<b>APPENDIX C VALUES FOR UNITS QUALIFIER.....</b>	<b>85</b>
	<b>APPENDIX D UML NOTATION .....</b>	<b>86</b>
	<b>APPENDIX E GLOSSARY.....</b>	<b>89</b>
	<b>APPENDIX F UNICODE USAGE .....</b>	<b>92</b>
F.1	MOF Text.....	92
F.2	Quoted Strings .....	92
	<b>APPENDIX G GUIDELINES.....</b>	<b>94</b>
G.1	Mapping of Octet Strings.....	94
G.2	SQL Reserved Words.....	94
	<b>APPENDIX H REFERENCES .....</b>	<b>97</b>

**Table of Figures**

FIGURE 1-1 FOUR WAYS TO USE CIM.....	3
FIGURE 2-1 META SCHEMA STRUCTURE.....	6
FIGURE 2-2 REFERENCE NAMING.....	8
FIGURE 2-3 REFERENCES, RANGES, AND DOMAINS.....	9
FIGURE 2-4 REFERENCES, RANGES, DOMAINS AND INHERITANCE.....	10
FIGURE 5-1 DEFINITIONS OF INSTANCES AND CLASSES.....	47
FIGURE 5-2 EXPORTING TO MOF.....	48
FIGURE 5-3 INFORMATION EXCHANGE.....	49
FIGURE 5-4 EXAMPLE OF WEAK ASSOCIATION.....	50
FIGURE 5-5 OBJECT NAMING.....	52
FIGURE 5-6 NAMESPACES.....	53
FIGURE 5-7 NAMESPACE PATH.....	56
FIGURE 5-8 PRAGMA EXAMPLE.....	57
FIGURE 5-9 NAMESPACE PATH EXAMPLE.....	58
FIGURE 5-10 REFERENCES BETWEEN MANAGEMENT SYSTEMS.....	59
FIGURE 5-11 EXAMPLE OF NONLOCAL QUALIFIER.....	60
FIGURE 6-1 TECHNIQUE MAPPING EXAMPLE.....	61
FIGURE 6-2 MIF TECHNIQUE MAPPING EXAMPLE.....	62
FIGURE 6-3 RECAST MAPPING.....	62
FIGURE 7-1 REPOSITORY PARTITIONS.....	66
FIGURE 7-2 HOMOGENEOUS AND HETEROGENEOUS EXPORT.....	68
FIGURE 7-3 SCRATCH PADS AND MAPPING.....	69

## 1 INTRODUCTION AND OVERVIEW

---

This section describes the many ways in which the Common Information Model (CIM) can be used. It provides a context in which the details described in the later chapters can be understood.

Ideally, information used to perform tasks is organized or structured to allow disparate groups of people to use it. This can be accomplished by developing a model or representation of the details required by people working within a particular domain. Such an approach can be referred to as an information model. An information model requires a set of legal statement types or syntax to capture the representation, and a collection of actual expressions necessary to manage common aspects of the domain (in this case, complex computer systems). Because of the focus on common aspects, the DMTF refers to this information model as CIM, the Common Information Model.

This document describes an object-oriented meta model based on the Unified Modeling Language (UML). This model includes expressions for common elements that must be clearly presented to management applications (for example, object classes, properties, methods and associations). This document does not describe specific CIM implementations, APIs, or communication protocols – those topics will be addressed in a future version of the specification. For information on the current core and common schemas developed using this meta model, contact the Distributed Management Task Force.

### 1.1 CIM Management Schema

Management schemas are the building blocks for management platforms and management applications, such as device configuration, performance management, and change management. CIM is structured in such a way that the managed environment can be seen as a collection of interrelated systems, each of which is composed of a number of discrete elements.

CIM supplies a set of classes with properties and associations that provide a well-understood conceptual framework within which it is possible to organize the available information about the managed environment. It is assumed that CIM will be clearly understood by any programmer required to write code that will operate against the object schema, or by any schema designer intending to make new information available within the managed environment.

CIM itself is structured into these distinct layers:

- Core model—an information model that captures notions that are applicable to all areas of management.
- Common model—an information model that captures notions that are common to particular management areas, but independent of a particular technology or implementation. The common areas are systems, applications, networks and devices. The information model is specific enough to provide a basis for the development of management applications. This schema provides a set of base classes for extension into the area of technology-specific schemas. The Core and Common models together are referred to in this document as the CIM schema.

- Extension schemas—represent technology-specific extensions of the Common model. These schemas are specific to environments, such as operating systems (for example, UNIX or Microsoft Windows).

### ***1.1.1 Core Model***

The Core model is a small set of classes, associations and properties that provide a basic vocabulary for analyzing and describing managed systems. The Core model represents a starting point for the analyst in determining how to extend the common schema. While it is possible that additional classes will be added to the Core model over time, major re-interpretations of the Core model classes are not anticipated.

### ***1.1.2 Common Model***

The Common model is a basic set of classes that define various technology-independent areas. The areas are systems, applications, networks and devices. The classes, properties, associations and methods in the Common model are intended to provide a view of the area that is detailed enough to use as a basis for program design and, in some cases, implementation. Extensions are added below the Common model in platform-specific additions that supply concrete classes and implementations of the Common model classes. As the Common model is extended, it will offer a broader range of information.

### ***1.1.3 Extension Schema***

The Extension schemas are technology-specific extensions to the Common model. It is expected that the Common model will evolve as a result of the promotion of objects and properties defined in the Extension schemas.

## **1.2 CIM Implementations**

CIM is a conceptual model that is not bound to a particular implementation. This allows it to be used to exchange management information in a variety of ways; four of these ways are illustrated in Figure 1-1. It is possible to use these ways in combination within a management application.